Android App Programming

Lecture 6: Android Runtime
December 5, 2017

Thomas Nowak
Université Paris-Sud
Last lecture recap
SQLite

- DBMS integrated into Android: SQLite
- packaged into the app
  - runs in the same process
  - no connection setup necessary
- use the class **SQLiteDatabase** included in Android
Android Runtime
Runtime

- runtime = execution environment of programs

- implements the computational model of the programming language

- ex: C calling conventions
  - call stack, parameter/return value passing, caller/callee saved registers

- ex: Python interpreter

- ex: Java Virtual Machine
else if(strncmp(op, "add", OP_LEN) == 0) {
    int *op1 = find_reg(arg1);
    int *op2 = find_reg(arg2);
    int *res = op1;
    *res = *op1 + *op2;
    program_counter++;
} else if(strncmp(op, "sub", OP_LEN) == 0) {
    int *op1 = find_reg(arg1);
    int *op2 = find_reg(arg2);
    int *res = op1;
    *res = *op1 - *op2;
    program_counter++;
}
else if(strcmp(op, "add") == 0)
{
    int *op1 = find_reg(arg1);
    int *op2 = find_reg(arg2);
    int *res = op1;

    *res = *op1 + *op2;

    program_counter++;
}
else if(strcmp(op, "sub") == 0)
{
    int *op1 = find_reg(arg1);
Dalvik and ART
JIT in ART

- **.oat files**
  - Using AOT Binary
  - ART

- **.dex files**
  - No AOT Binary
  - ART
    - cold code
  - ART
    - hot code
    - Just-in-Time (JIT)

- Interpreter
- App Runs
Stack vs Heap

- variables on the stack: implicitly destroyed after method return (pop of stack frame)

- variables/constantes on the heap: destruction isn’t implicit; no temporal or logical ordering between objects on the heap

- error source in C++: managing the heap, in particular failure to destroy unused objects (memory leak)

- alternative of manually managing the heap: automatic garbage collection by the runtime
Garbage Collection

- garbage collection = automatic identification and of unused objects
- basic mechanism: identification of unused object based on active references
- choice of moment of collection
- goal: don’t block execution of the program
- additional challenge: avoid fragmentation of memory
Reference Counting

- idea: count the number of references to every object
- if count = 0, then destroy the object
- destroying one object can (transitively) cause destruction of a large number of other objects; problematic if immediate destruction
- problem with cyclic references
Mark-and-Sweep

• mark: calculate successors in the directed graph of objects and their references

• sweep: destroy all non-marked objects

• moment of collection:
  • stop-the-world
  • concurrent
Android Runtime

• originally Dalvik virtual machine

• now ART w/ AOT and JIT compilation of .dex files to native code

• runtime includes garbage collector (even outside of VM)
Advance Reading

• Next week’s topic: Fragments
• Chapter 29
Questions